Lost in Translation: A Study of Bugs Introduced by Large Language Models while Translating Code

Rangeet Pan[*1], Ali Reza Ibrahimzada[*Ψ2], Rahul Krishna[1], Divya Sankar[1], Lambert Pouguem Wassi[1], Michele Merler[1], Boris Sobolev[1], Raju Pavuluri[1], Saurabh Sinha[1], Reyhaneh Jabbarvand[2]

[1]IBM Research [2]University of Illinois Urbana-Champaign

Work done as part of IBM-Illinois Discovery Accelerator Institute (IIDAI)

[Ψ]*Work done when **Ali** was an intern at IBM Research*

Leaderboard: codetlingua.github.io

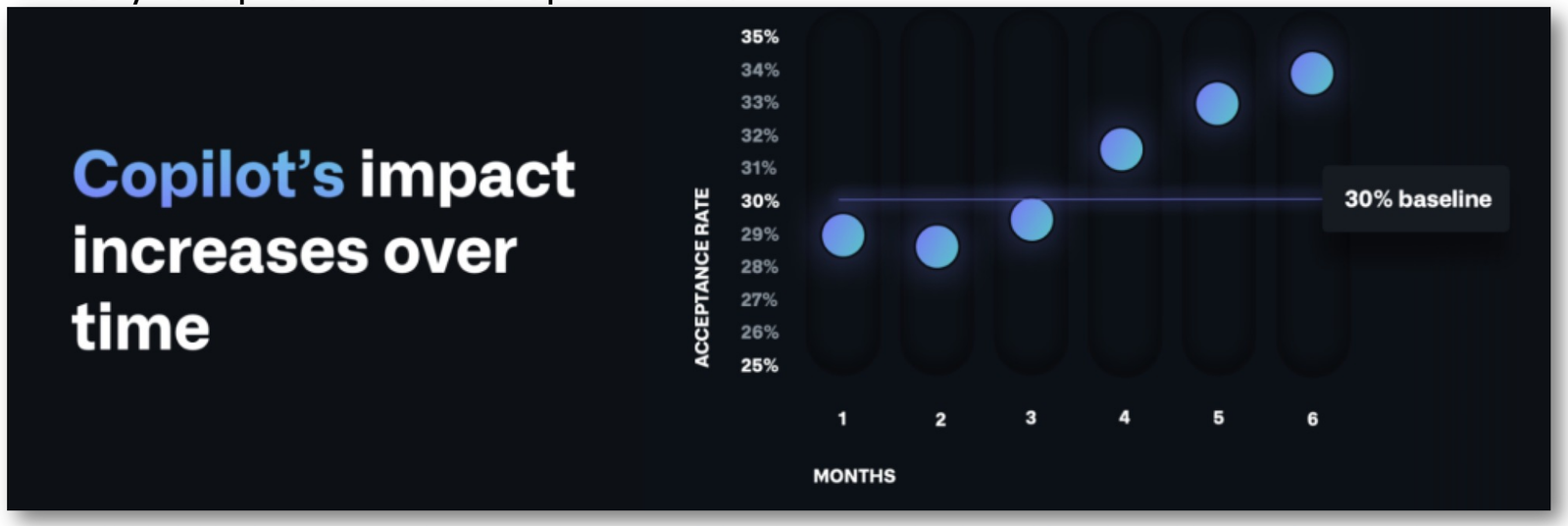
INTELLIGENT Code Analysis Testing

[*]Equal Contribution

LLMs for Code Generation & Understanding

- LLMs are used for assisting developers
  - **2021 & 2022:** Codex, CodeT5, GitHub Copilot, AlphaCode, etc.
  - **2023 & 2024:** GPT-4, Gemini, CodeLlama, Claude 3, etc.

Today, GitHub Copilot has been activated by more than one million developers and adopted by over 20,000 organizations. It has generated over three billion accepted lines of code, and is the world's most widely adopted AI developer tool.



https://github.blog/2023-06-27-the-economic-impact-of-the-ai-powered-developer-lifecycle-and-lessons-from-github-copilot/

What is Code Translation?

Code translation converts source code from one programming language to another

What is Code Translation?

Code translation converts source code from one programming language to another

Java Code

```java
public class Main {
    public static void main(String[] args) {
        int max = 5;
        for (int i = 0; i < max; i++) {
            System.out.println(i);
        }
    }
}
```

What is Code Translation?

Code translation converts source code from one programming language to another

Java Code

```java
public class Main {
    public static void main(String[] args) {
        int max = 5;
        for (int i = 0; i < max; i++) {
            System.out.println(i);
        }
    }
}
```

Python Code

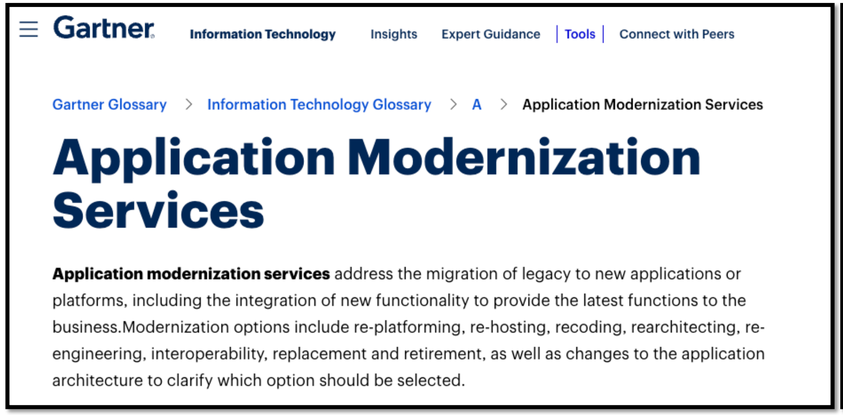
```python
max = 5
for i in range(max):
    print(i)
```

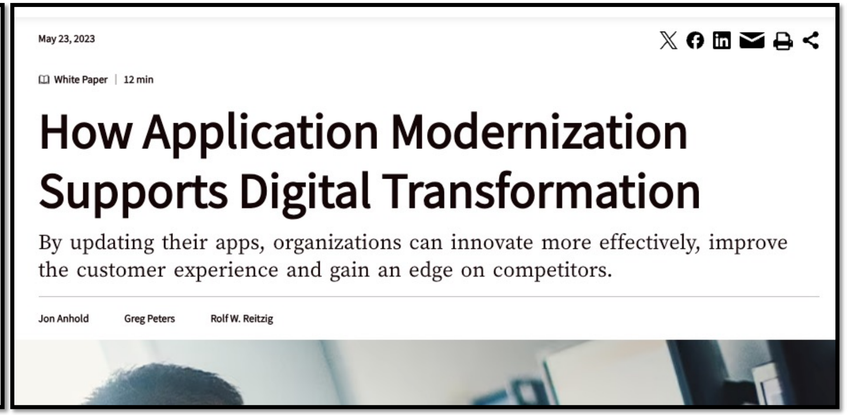
Why Code Translation?

❖ Use cases of code translation:

Why Code Translation?

- Use cases of code translation:
  - Application modernization
  - Migrating legacy software to cloud-native applications





Why Code Translation?

- Use cases of code translation:
  - Application modernization
  - Migrating legacy software to cloud-native applications
- Code translation is challenging as it requires understanding both syntax and semantics

Why Code Translation?

- Use cases of code translation:
  - Application modernization
  - Migrating legacy software to cloud-native applications
- Code translation is challenging as it requires understanding both syntax and semantics
  - Transpiler techniques, e.g., C2Rust, CxGO, Java2C#
  - LLMs **`generalizes`** well and generates more **`natural code`**

Why Code Translation?

- Use cases of code translation:
  - Application modernization
  - Migrating legacy software to cloud-native applications
- Code translation is challenging as it requires understanding both syntax and semantics
  - Transpiler techniques, e.g., C2Rust, CxGO, Java2C#
  - LLMs **generalizes** well and generates more **natural code**
- Existing learning-based techniques use small models (**<1B** parameter) and fail to achieve high accuracy

An Empirical Study on LLM-based Code Translation

- We study the limitations of code translation across multiple **PLs**, **benchmarks**, and **real-life projects** using various **general** and **code LLMs**

An Empirical Study on LLM-based Code Translation

- We study the limitations of code translation across multiple **PLs**, **benchmarks**, and **real-life projects** using various **general** and **code LLMs**
- Effectiveness of LLMs in code translation

An Empirical Study on LLM-based Code Translation

- We study the limitations of code translation across multiple **PLs**, **benchmarks**, and **real-life projects** using various **general** and **code LLMs**
- Effectiveness of LLMs in code translation
- Taxonomy of LLM translation bugs

An Empirical Study on LLM-based Code Translation

- We study the limitations of code translation across multiple **PLs**, **benchmarks**, and **real-life projects** using various **general** and **code LLMs**
- Effectiveness of LLMs in code translation
- Taxonomy of LLM translation bugs
- Comparison with non-LLM-based approaches (aka transpilers)

An Empirical Study on LLM-based Code Translation

- We study the limitations of code translation across multiple **PLs**, **benchmarks**, and **real-life projects** using various **general** and **code LLMs**
- Effectiveness of LLMs in code translation
- Taxonomy of LLM translation bugs
- Comparison with non-LLM-based approaches (aka transpilers)
- Mitigating translation bugs

Empirical Setup

❖ Subject LLMs:

- We aim at using models **<20B** parameters due to compute constraints
- SOTA LLMs at the time of study

Modality	**Code**			**Text**			
Subject LLMs	CodeGen	CodeGeeX	StarCoder	GPT-4	Llama 2	TB-Airoboros	TB-Vicuna
Size	16B	13B	15.5B	-	13B	13B	13B
Context Window	2,048	2,048	2,048	8,192	4,096	2,048	2,048
Release Date	May 2023	March 2023	May 2023	March 2023	July 2023	May 2023	May 2023

Empirical Setup

❖ Subject LLMs:

- We aim at using models **<20B** parameters due to compute constraints
- SOTA LLMs at the time of study

Modality	**Code**			**Text**			
Subject LLMs	CodeGen	CodeGeeX	StarCoder	GPT-4	Llama 2	TB-Airoboros	TB-Vicuna
Size	16B	13B	15.5B	-	13B	13B	13B
Context Window	2,048	2,048	2,048	8,192	4,096	2,048	2,048
Release Date	May 2023	March 2023	May 2023	March 2023	July 2023	May 2023	May 2023

Empirical Setup

❖ Subject LLMs:

- We aim at using models **<20B** parameters due to compute constraints
- SOTA LLMs at the time of study

Modality	**Code**			**Text**			
Subject LLMs	CodeGen	CodeGeeX	StarCoder	GPT-4	Llama 2	TB-Airoboros	TB-Vicuna
Size	16B	13B	15.5B	-	13B	13B	13B
Context Window	2,048	2,048	2,048	8,192	4,096	2,048	2,048
Release Date	May 2023	March 2023	May 2023	March 2023	July 2023	May 2023	May 2023

Empirical Setup

❖ Subject LLMs:

Leaderboard: codetlingua.github.io

- We aim at using models **<20B** parameters due to compute constraints
- SOTA LLMs at the time of study

Modality	**Code**			**Text**			
Subject LLMs	CodeGen	CodeGeeX	StarCoder	GPT-4	Llama 2	TB-Airoboros	TB-Vicuna
Size	16B	13B	15.5B	-	13B	13B	13B
Context Window	2,048	2,048	2,048	8,192	4,096	2,048	2,048
Release Date	May 2023	March 2023	May 2023	March 2023	July 2023	May 2023	May 2023

Empirical Setup

❖ Subject LLMs:

Leaderboard: codetlingua.github.io

- We aim at using models **<20B** parameters due to compute constraints
- SOTA LLMs at the time of study

Modality	**Code**			**Text**			
Subject LLMs	CodeGen	CodeGeeX	StarCoder	GPT-4	Llama 2	TB-Airoboros	TB-Vicuna
Size	16B	13B	15.5B	-	13B	13B	13B
Context Window	2,048	2,048	2,048	8,192	4,096	2,048	2,048
Release Date	May 2023	March 2023	May 2023	March 2023	July 2023	May 2023	May 2023

❖ Subject PLs:

- TIOBE Score[1], different paradigms, and availability of datasets
- C, C++, Go, Java, Python

[1] https://www.tiobe.com/tiobe-index/

Effectiveness of LLMs in Code Translation

Crafted benchmark datasets and real-life projects

Dataset	Source Language	Source Samples	# Tests	Target Language	#Translations
CodeNet[1]	C	200	200	C++,Go,Java,Python	800
	C++	200	200	C,Go,Java,Python	800
	Go	200	200	C,C++,Java,Python	800
	Java	200	200	C,C++,Go,Python	800
	Python	200	200	C,C++,Go,Java	800
Total / Average (CodeNet)	-	1,000	1,000	-	4,000
AVATAR[2]	Java	249	6,255	C,C++,Go,Python	996
	Python	250		C,C++,Go,Java	1,000
Total / Average (AVATAR)	-	499	6,255	-	1,996
Evalplus[3]	Python	164	2,682	Java	164
Commons-CLI[4]	Java	22	310	Python	22
Click[5]	Python	15	611	Java	15
Total / Average (All)	-	1,700	10,858	-	6,197

[1]https://github.com/IBM/Project_CodeNet [2]https://github.com/wasiahmad/AVATAR [3]https://github.com/evalplus/evalplus
[4]https://github.com/apache/commons-cli [5]https://click.palletsprojects.com/en/8.1.x/

Effectiveness of LLMs in Code Translation

Crafted benchmark datasets and real-life projects

Dataset	Source Language	Source Samples	# Tests	Target Language	#Translations
CodeNet[1]	C	200	200	C++,Go,Java,Python	800
	C++	200	200	C,Go,Java,Python	800
	Go	200	200	C,C++,Java,Python	800
	Java	200	200	C,C++,Go,Python	800
	Python	200	200	C,C++,Go,Java	800
Total / Average (CodeNet)	-	1,000	1,000	-	4,000
AVATAR[2]	Java	249	6,255	C,C++,Go,Python	996
	Python	250		C,C++,Go,Java	1,000
Total / Average (AVATAR)	-	499	6,255	-	1,996
Evalplus[3]	Python	164	2,682	Java	164
Commons-CLI[4]	Java	22	310	Python	22
Click[5]	Python	15	611	Java	15
Total / Average (All)	-	1,700	10,858	-	6,197

[1]https://github.com/IBM/Project_CodeNet [2]https://github.com/wasiahmad/AVATAR [3]https://github.com/evalplus/evalplus
[4]https://github.com/apache/commons-cli [5]https://click.palletsprojects.com/en/8.1.x/

Effectiveness of LLMs in Code Translation

Crafted benchmark datasets and real-life projects

Dataset	Source Language	Source Samples	# Tests	Target Language	#Translations
CodeNet[1]	C	200	200	C++,Go,Java,Python	800
	C++	200	200	C,Go,Java,Python	800
	Go	200	200	C,C++,Java,Python	800
	Java	200	200	C,C++,Go,Python	800
	Python	200	200	C,C++,Go,Java	800
Total / Average (CodeNet)	-	1,000	1,000	-	4,000
AVATAR[2]	Java	249	6,255	C,C++,Go,Python	996
	Python	250		C,C++,Go,Java	1,000
Total / Average (AVATAR)	-	499	6,255	-	1,996
Evalplus[3]	Python	164	2,682	Java	164
Commons-CLI[4]	Java	22	310	Python	22
Click[5]	Python	15	611	Java	15
Total / Average (All)	-	1,700	10,858	-	6,197

[1]https://github.com/IBM/Project_CodeNet [2]https://github.com/wasiahmad/AVATAR [3]https://github.com/evalplus/evalplus
[4]https://github.com/apache/commons-cli [5]https://click.palletsprojects.com/en/8.1.x/

Effectiveness of LLMs in Code Translation

Crafted benchmark datasets and real-life projects

Dataset	Source Language	Source Samples	# Tests	Target Language	#Translations
CodeNet[1]	C	200	200	C++,Go,Java,Python	800
	C++	200	200	C,Go,Java,Python	800
	Go	200	200	C,C++,Java,Python	800
	Java	200	200	C,C++,Go,Python	800
	Python	200	200	C,C++,Go,Java	800
Total / Average (CodeNet)	-	1,000	1,000	-	4,000
AVATAR[2]	Java	249	6,255	C,C++,Go,Python	996
	Python	250		C,C++,Go,Java	1,000
Total / Average (AVATAR)	-	499	6,255	-	1,996
Evalplus[3]	Python	164	2,682	Java	164
Commons-CLI[4]	Java	22	510	Python	22
Click[5]	Python	15	611	Java	15
Total / Average (All)	-	1,700	10,858	-	6,197

[1]https://github.com/IBM/Project_CodeNet [2]https://github.com/wasiahmad/AVATAR [3]https://github.com/evalplus/evalplus
[4]https://github.com/apache/commons-cli [5]https://click.palletsprojects.com/en/8.1.x/

Effectiveness of LLMs in Code Translation

Crafted benchmark datasets and real-life projects

Dataset	Source Language	Source Samples	# Tests	Target Language	#Translations
CodeNet[1]	C	200	200	C++,Go,Java,Python	800
	C++	200	200	C,Go,Java,Python	800
	Go	200	200	C,C++,Java,Python	800
	Java	200	200	C,C++,Go,Python	800
	Python	200	200	C,C++,Go,Java	800
Total / Average (CodeNet)	-	1,000	1,000	-	4,000
AVATAR[2]	Java	249	6,255	C,C++,Go,Python	996
	Python	250		C,C++,Go,Java	1,000
Total / Average (AVATAR)	-	499	6,255	-	1,996
Evalplus[3]	Python	164	2,682	Java	164
Commons-CLI[4]	Java	22	310	Python	22
Click[5]	Python	15	611	Java	15
Total / Average (All)	-	1,700	10,858	-	6,197

[1]https://github.com/IBM/Project_CodeNet [2]https://github.com/wasiahmad/AVATAR [3]https://github.com/evalplus/evalplus
[4]https://github.com/apache/commons-cli [5]https://click.palletsprojects.com/en/8.1.x/

Effectiveness of LLMs in Code Translation

How do state-of-the-art general and code LLMs perform in code translation across various benchmarks?

Dataset	Source Language	% Successful Translations						
		CodeGen	CodeGeeX	StarCoder	GPT-4	Llama 2	TB-Airoboros	TB-Vicuna
CodeNet	C	23.4%	14.9%	42.0%	83.0%	14.9%	18.8%	4.4%
	C++	14.0%	3.6%	39.1%	80.0%	9.5%	8.3%	3.4%
	Go	14.3%	5.9%	42.0%	85.5%	16.9%	6.6%	0.9%
	Java	21.3%	10.3%	30.3%	81.3%	13.9%	6.5%	0.1%
	Python	17.5%	7.3%	33.3%	79.9%	11.0%	6.5%	1.0%
Total / Average (CodeNet)	-	18.1%	8.4%	37.3%	82.0%	13.2%	9.3%	2.0%
AVATAR	Java	8.1%	1.8%	11.9%	70.8%	1.8%	5.0%	0.0%
	Python	3.8%	1.6%	14.2%	52.2%	4.7%	0.9%	0.9%
Total / Average (AVATAR)	-	5.9%	1.7%	13.0%	61.5%	3.2%	3.0%	0.4%
Evalplus	Python	16.5%	3.7%	22.0%	79.3%	1.2%	14.0%	7.9%
Commons-CLI	Java	0.0%	0.0%	0.0%	13.6%	0.0%	0.0%	0.0%
Click	Python	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Total / Average (All)	-	8.1%	2.8%	14.5%	47.3%	3.5%	5.3%	2.1%

Effectiveness of LLMs in Code Translation

How do state-of-the-art general and code LLMs perform in code translation across various benchmarks?

Dataset	Source Language	% Successful Translations						
		CodeGen	CodeGeeX	StarCoder	GPT-4	Llama 2	TB-Airoboros	TB-Vicuna
CodeNet	C	23.4%	14.9%	42.0%	83.0%	14.9%	18.8%	4.4%
	C++	14.0%	3.6%	39.1%	80.0%	9.5%	8.3%	3.4%
	Go	14.3%	5.9%	42.0%	85.5%	16.9%	6.6%	0.9%
	Java	21.3%	10.3%	30.3%	81.3%	13.9%	6.5%	0.1%
	Python	17.5%	7.3%	33.3%	79.9%	11.0%	6.5%	1.0%
Total / Average (CodeNet)	-	18.1%	8.4%	37.3%	82.0%	13.2%	9.3%	2.0%
AVATAR	Java	8.1%	1.8%	11.9%	70.8%	1.8%	5.0%	0.0%
	Python	3.8%	1.6%	14.2%	52.2%	4.7%	0.9%	0.9%
Total / Average (AVATAR)	-	5.9%	1.7%	13.0%	61.5%	3.2%	3.0%	0.4%
Evalplus	Python	16.5%	3.7%	22.0%	79.3%	1.2%	14.0%	7.9%
Commons-CLI	Java	0.0%	0.0%	0.0%	13.6%	0.0%	0.0%	0.0%
Click	Python	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Total / Average (All)	-	8.1%	2.8%	14.5%	47.3%	3.5%	5.3%	2.1%

Effectiveness of LLMs in Code Translation

How do state-of-the-art general and code LLMs perform in code translation across various benchmarks?

Dataset	Source Language	% Successful Translations						
		CodeGen	CodeGeeX	StarCoder	GPT-4	Llama 2	TB-Airoboros	TB-Vicuna
CodeNet	C	23.4%	14.9%	42.0%	83.0%	14.9%	18.8%	4.4%
	C++	14.0%	3.6%	39.1%	80.0%	9.5%	8.3%	3.4%
	Go	14.3%	5.9%	42.0%	85.5%	16.9%	6.6%	0.9%
	Java	21.3%	10.3%	30.3%	81.3%	13.9%	6.5%	0.1%
	Python	17.5%	7.3%	33.3%	79.9%	11.0%	6.5%	1.0%
Total / Average (CodeNet)	-	18.1%	8.4%	37.3%	82.0%	13.2%	9.3%	2.0%
AVATAR	Java	8.1%	1.8%	11.9%	70.8%	1.8%	5.0%	0.0%
	Python	3.8%	1.6%	14.2%	52.2%	4.7%	0.9%	0.9%
Total / Average (AVATAR)	-	5.9%	1.7%	13.0%	61.5%	3.2%	3.0%	0.4%
Evalplus	Python	16.5%	3.7%	22.0%	79.3%	1.2%	14.0%	7.9%
Commons-CLI	Java	0.0%	0.0%	0.0%	13.6%	0.0%	0.0%	0.0%
Click	Python	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Total / Average (All)	-	8.1%	2.8%	14.5%	47.3%	3.5%	5.3%	2.1%

Effectiveness of LLMs in Code Translation

How do state-of-the-art general and code LLMs perform in code translation across various benchmarks?

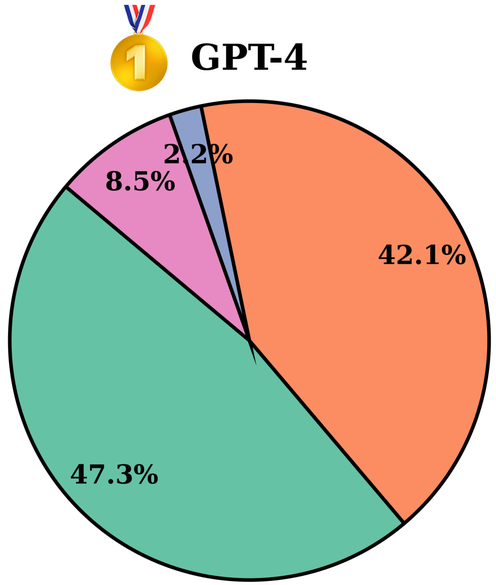
Dataset	Source Language	% Successful Translations						
		CodeGen	CodeGeeX	StarCoder	GPT-4	Llama 2	TB-Airoboros	TB-Vicuna
CodeNet	C	23.4%	14.9%	42.0%	83.0%	14.9%	18.8%	4.4%
	C++	14.0%	3.6%	39.1%	80.0%	9.5%	8.3%	3.4%
	Go	14.3%	5.9%	42.0%	85.5%	16.9%	6.6%	0.9%
	Java	21.3%	10.3%	30.3%	81.3%	13.9%	6.5%	0.1%
	Python	17.5%	7.3%	33.3%	79.9%	11.0%	6.5%	1.0%
Total / Average (CodeNet)	-	18.1%	8.4%	37.3%	82.0%	13.2%	9.3%	2.0%
AVATAR	Java	8.1%	1.8%	11.9%	70.8%	1.8%	5.0%	0.0%
	Python	3.8%	1.6%	14.2%	52.2%	4.7%	0.9%	0.9%
Total / Average (AVATAR)	-	5.9%	1.7%	13.0%	61.5%	3.2%	3.0%	0.4%
Evalplus	Python	16.5%	3.7%	22.0%	79.3%	1.2%	14.0%	7.9%
Commons-CLI	Java	0.0%	0.0%	0.0%	13.6%	0.0%	0.0%	0.0%
Click	Python	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Total / Average (All)	-	8.1%	2.8%	14.5%	47.3%	3.5%	5.3%	2.1%

Outcome of Unsuccessful Translation

What are the outcomes of unsuccessful translations: errors due to syntactic incorrectness (Compile)? or semantic? (Runtime, Functional, etc.)

Outcome of Unsuccessful Translation

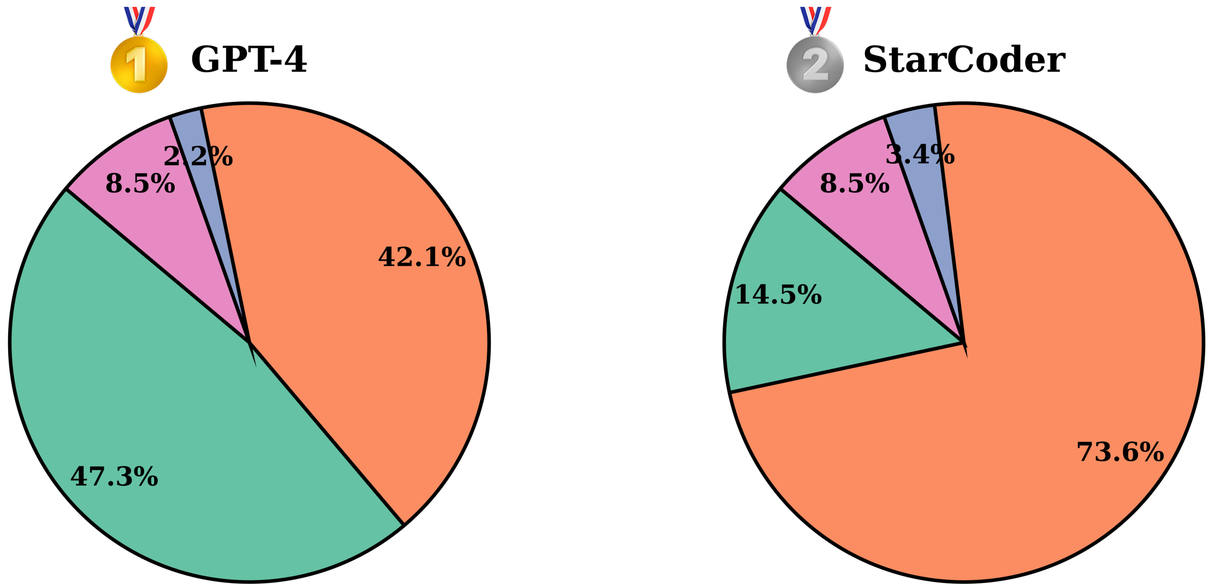
What are the outcomes of unsuccessful translations: errors due to syntactic incorrectness (Compile)? or semantic? (Runtime, Functional, etc.)



Correct	Compile Error	Runtime Error	Wrong Output

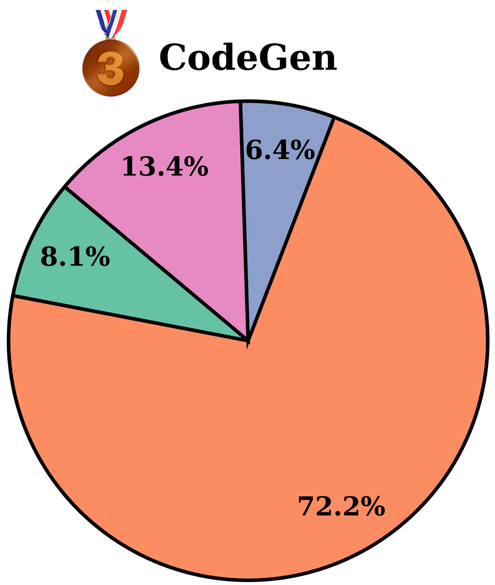
Outcome of Unsuccessful Translation

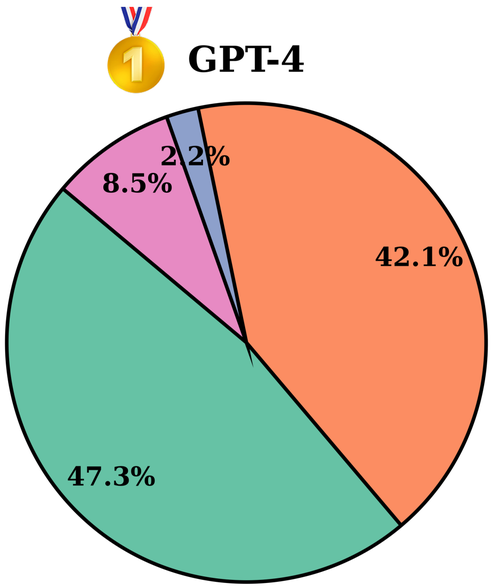
What are the outcomes of unsuccessful translations: errors due to syntactic incorrectness (Compile)? or semantic? (Runtime, Functional, etc.)

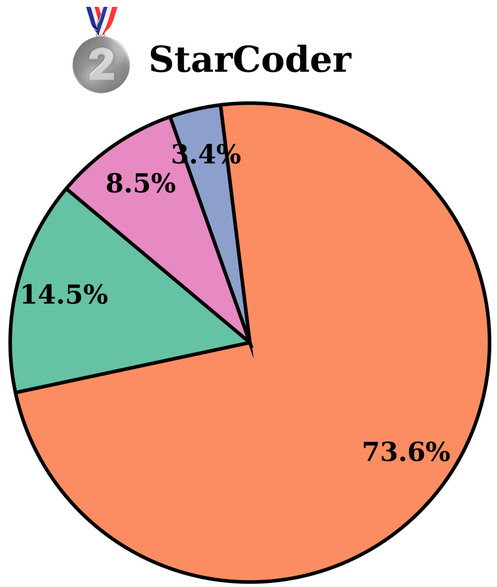


Outcome of Unsuccessful Translation

What are the outcomes of unsuccessful translations: errors due to syntactic incorrectness (Compile)? or semantic? (Runtime, Functional, etc.)







Correct	Compile Error	Runtime Error	Wrong Output

Taxonomy of Translation Bugs 

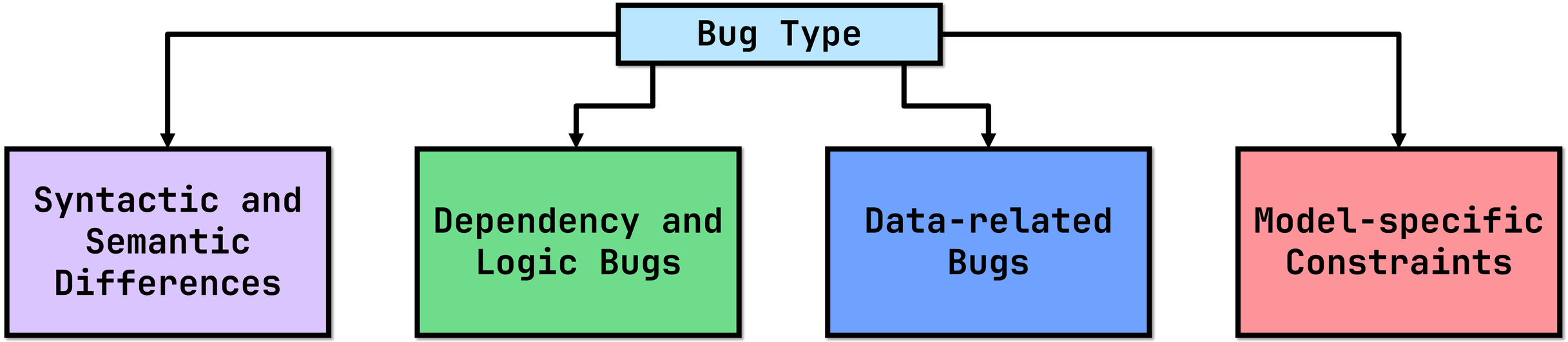
- We manually investigated and root caused **1,748** unsuccessful translations from the best performing model (GPT-4)

Taxonomy of Translation Bugs 

- We manually investigated and root caused **1,748** unsuccessful translations from the best performing model (GPT-4)
  - **8** human labelers (industry researchers and software engineers): **630** person-hours
  - **15** unique categories of translation bugs (details in paper)

Taxonomy of Translation Bugs

- We manually investigated and root caused **1,748** unsuccessful translations from the best performing model (GPT-4)
  - **8** human labelers (industry researchers and software engineers): **630** person-hours
  - **15** unique categories of translation bugs (details in paper)



Syntactic and Semantic Differences b/w PLs

LLMs fail to properly handle syntactic and semantic differences between PLs

Duplicating source syntax to target language

```
constant ld PI = atan2l(0, -1); # Original C++ code
PI = atan2l(0, -1)              # Incorrect Go code
```

Syntactic and Semantic Differences b/w PLs

LLMs fail to properly handle syntactic and semantic differences between PLs

Duplicating source syntax to target language

```
constant ld PI = atan2l(0, -1); # Original C++ code
PI = atan2l(0, -1)              # Incorrect Go code
```

🚨 **atan2l is not available in target language (Go)** 🚨

Dependency and Logic Bugs

These bugs pertain to incorrect dependencies and logic of translated code

Missing library imports and definitions

```
void solve(){...}                  # Original C++ code
signed main(){...solve();}         # Incorrect Go code
```

Dependency and Logic Bugs

These bugs pertain to incorrect dependencies and logic of translated code

Missing library imports and definitions

```
void solve(){...}                  # Original C++ code
signed main(){...solve();}         # Incorrect Go code
```

🚨 **solve() is called but not implemented in Go** 🚨

Data-related Bugs

LLMs suffer translating code which reads and writes data to stdin / stdout

Output Formatting

```
System.out.print("H"); System.out.println(Y - 1); # Java code
print("H", Y - 1)                                  # Python code
```

Data-related Bugs

LLMs suffer translating code which reads and writes data to stdin / stdout

Output Formatting

```
System.out.print("H"); System.out.println(Y - 1); # Java code
print("H", Y - 1)                                  # Python code
```

🚨 **extra space added in Python output** 🚨

Translation Bugs in Real-World Projects

🚨**WE ARE THE FIRST EVER TO TRANSLATE AND INVESTIGATE REAL-WORLD PROJECTS**🚨

Real-world applications pose more complex challenges for code translation, such as handling method overloading, exceptions, inheritance relations, etc.

Translation Bugs in Real-World Projects

🚨 **WE ARE THE FIRST EVER TO TRANSLATE AND INVESTIGATE REAL-WORLD PROJECTS** 🚨

Real-world applications pose more complex challenges for code translation, such as handling method overloading, exceptions, inheritance relations, etc.

Source Code (Java)

```java
public Options addOption(final Option opt) { ... }
public Options addOption(final String opt, final boolean hasArg,
                         final String desc) { ... }
public Options addOption(final String opt, final String desc) { ... }
```

Translation Bugs in Real-World Projects

🚨 **WE ARE THE FIRST EVER TO TRANSLATE AND INVESTIGATE REAL-WORLD PROJECTS** 🚨

Real-world applications pose more complex challenges for code translation, such as handling method overloading, exceptions, inheritance relations, etc.

Source Code (Java)

```java
public Options addOption(final Option opt) { ... }
public Options addOption(final String opt, final boolean hasArg,
                         final String desc) { ... }
public Options addOption(final String opt, final String desc) { ... }
```

Translated Code (Python)

```python
def add_option(self, opt): ...
def add_option_arg(self, opt, hasArg, desc): ...
def add_option_desc(self, opt, desc): ...
```

Feedback-based Prompt Crafting

A technique for providing additional contextual information in the prompt to enhance LLMs’ responses

Feedback-based Prompt Crafting

A technique for providing additional contextual information in the prompt to enhance LLMs' responses

```
You were asked to translate the following
$SRC_LANG code to $TRG_LANG:
$SRC_CODE
- - - - - - - - - - - - - - - - - - - - - -
Your response was the following $TRG_LANG code:
$INCORRECT_CODE
Executing your generated code gives the following
output:
${STACK_TRACE | ERROR_LOG | INCORRECT_OUTPUT}
- - - - - - - - - - - - - - - - - - - - - -
Can you re-generate your response and translate
the above $SRC_LANG code to $TRG_LANG. Do not add
any natural language description in your response.
- - - - - - - - - - - - - - - - - - - - - -
Your generated $TRG_LANG code should pass the
following test:
Test Input: $TEST_INPUT
Expected Output: $TEST_OUTPUT
- - - - - - - - - - - - - - - - - - - - - -
$TRG_LANG Code: // Generated Code
```

1 2 3 4 5

V: Vanilla **F: Feedback**

Feedback-based Prompt Crafting

A technique for providing additional contextual information in the prompt to enhance LLMs' responses

```
You were asked to translate the following
$SRC_LANG code to $TRG_LANG:
$SRC_CODE
- - - - - - - - - - - - - - - - - - - - - -
Your response was the following $TRG_LANG code:
$INCORRECT_CODE
Executing your generated code gives the following
output:
${STACK_TRACE | ERROR_LOG | INCORRECT_OUTPUT}
- - - - - - - - - - - - - - - - - - - - - -
Can you re-generate your response and translate
the above $SRC_LANG code to $TRG_LANG. Do not add
any natural language description in your response.
- - - - - - - - - - - - - - - - - - - - - -
Your generated $TRG_LANG code should pass the
following test:
Test Input: $TEST_INPUT
Expected Output: $TEST_OUTPUT
- - - - - - - - - - - - - - - - - - - - - -
$TRG_LANG Code: // Generated Code
```

1. Remind the model about the task and source code (V)
2.
3.
4.
5.

(V): **Vanilla** (F): **Feedback**

Feedback-based Prompt Crafting

A technique for providing additional contextual information in the prompt to enhance LLMs' responses

```
You were asked to translate the following
$SRC_LANG code to $TRG_LANG:
$SRC_CODE
- - - - - - - - - - - - - - - - - - - - - -
Your response was the following $TRG_LANG code:
$INCORRECT_CODE
Executing your generated code gives the following
output:
${STACK_TRACE | ERROR_LOG | INCORRECT_OUTPUT}
- - - - - - - - - - - - - - - - - - - - - -
Can you re-generate your response and translate
the above $SRC_LANG code to $TRG_LANG. Do not add
any natural language description in your response.
- - - - - - - - - - - - - - - - - - - - - -
Your generated $TRG_LANG code should pass the
following test:
Test Input: $TEST_INPUT
Expected Output: $TEST_OUTPUT
- - - - - - - - - - - - - - - - - - - - - -
$TRG_LANG Code: // Generated Code
```

1. Remind the model about the task and source code (V)
2. Provide incorrect translation and feedback (F)
3.
4.
5.

(V): **Vanilla** (F): **Feedback**

Feedback-based Prompt Crafting

A technique for providing additional contextual information in the prompt to enhance LLMs' responses

```
You were asked to translate the following
$SRC_LANG code to $TRG_LANG:
$SRC_CODE
- - - - - - - - - - - - - - - - - - - - - - -
Your response was the following $TRG_LANG code:
$INCORRECT_CODE
Executing your generated code gives the following
output:
${STACK_TRACE | ERROR_LOG | INCORRECT_OUTPUT}
- - - - - - - - - - - - - - - - - - - - - - -
Can you re-generate your response and translate
the above $SRC_LANG code to $TRG_LANG. Do not add
any natural language description in your response.
- - - - - - - - - - - - - - - - - - - - - - -
Your generated $TRG_LANG code should pass the
following test:
Test Input: $TEST_INPUT
Expected Output: $TEST_OUTPUT
- - - - - - - - - - - - - - - - - - - - - - -
$TRG_LANG Code: // Generated Code
```

1. Remind the model about the task and source code (V)
2. Provide incorrect translation and feedback (F)
3. Determine the objective and constraints (F)
4.
5.

(V): **Vanilla** (F): **Feedback**

Feedback-based Prompt Crafting

A technique for providing additional contextual information in the prompt to enhance LLMs' responses

```
You were asked to translate the following
$SRC_LANG code to $TRG_LANG:
$SRC_CODE
- - - - - - - - - - - - - - - - - - - - - - - -
Your response was the following $TRG_LANG code:
$INCORRECT_CODE
Executing your generated code gives the following
output:
${STACK_TRACE | ERROR_LOG | INCORRECT_OUTPUT}
- - - - - - - - - - - - - - - - - - - - - - - -
Can you re-generate your response and translate
the above $SRC_LANG code to $TRG_LANG. Do not add
any natural language description in your response.
- - - - - - - - - - - - - - - - - - - - - - - -
Your generated $TRG_LANG code should pass the
following test:
Test Input: $TEST_INPUT
Expected Output: $TEST_OUTPUT
- - - - - - - - - - - - - - - - - - - - - - - -
$TRG_LANG Code: // Generated Code
```

1. Remind the model about the task and source code (V)
2. Provide incorrect translation and feedback (F)
3. Determine the objective and constraints (F)
4. A sample test I/O to further enrich the feedback (F)
5.

V: Vanilla **F: Feedback**

Feedback-based Prompt Crafting

A technique for providing additional contextual information in the prompt to enhance LLMs' responses

```
You were asked to translate the following
$SRC_LANG code to $TRG_LANG:
$SRC_CODE
- - - - - - - - - - - - - - - - - - - - - -
Your response was the following $TRG_LANG code:
$INCORRECT_CODE
Executing your generated code gives the following
output:
${STACK_TRACE | ERROR_LOG | INCORRECT_OUTPUT}
- - - - - - - - - - - - - - - - - - - - - -
Can you re-generate your response and translate
the above $SRC_LANG code to $TRG_LANG. Do not add
any natural language description in your response.
- - - - - - - - - - - - - - - - - - - - - -
Your generated $TRG_LANG code should pass the
following test:
Test Input: $TEST_INPUT
Expected Output: $TEST_OUTPUT
- - - - - - - - - - - - - - - - - - - - - -
$TRG_LANG Code: // Generated Code
```

1. Remind the model about the task and source code (V)
2. Provide incorrect translation and feedback (F)
3. Determine the objective and constraints (F)
4. A sample test I/O to further enrich the feedback (F)
5. Model-specific keyword (V)

(V): Vanilla **(F): Feedback**

Iterative Translation Bug Mitigation

- We iteratively prompt LLMs with additional feedback to repair unsuccessful cases

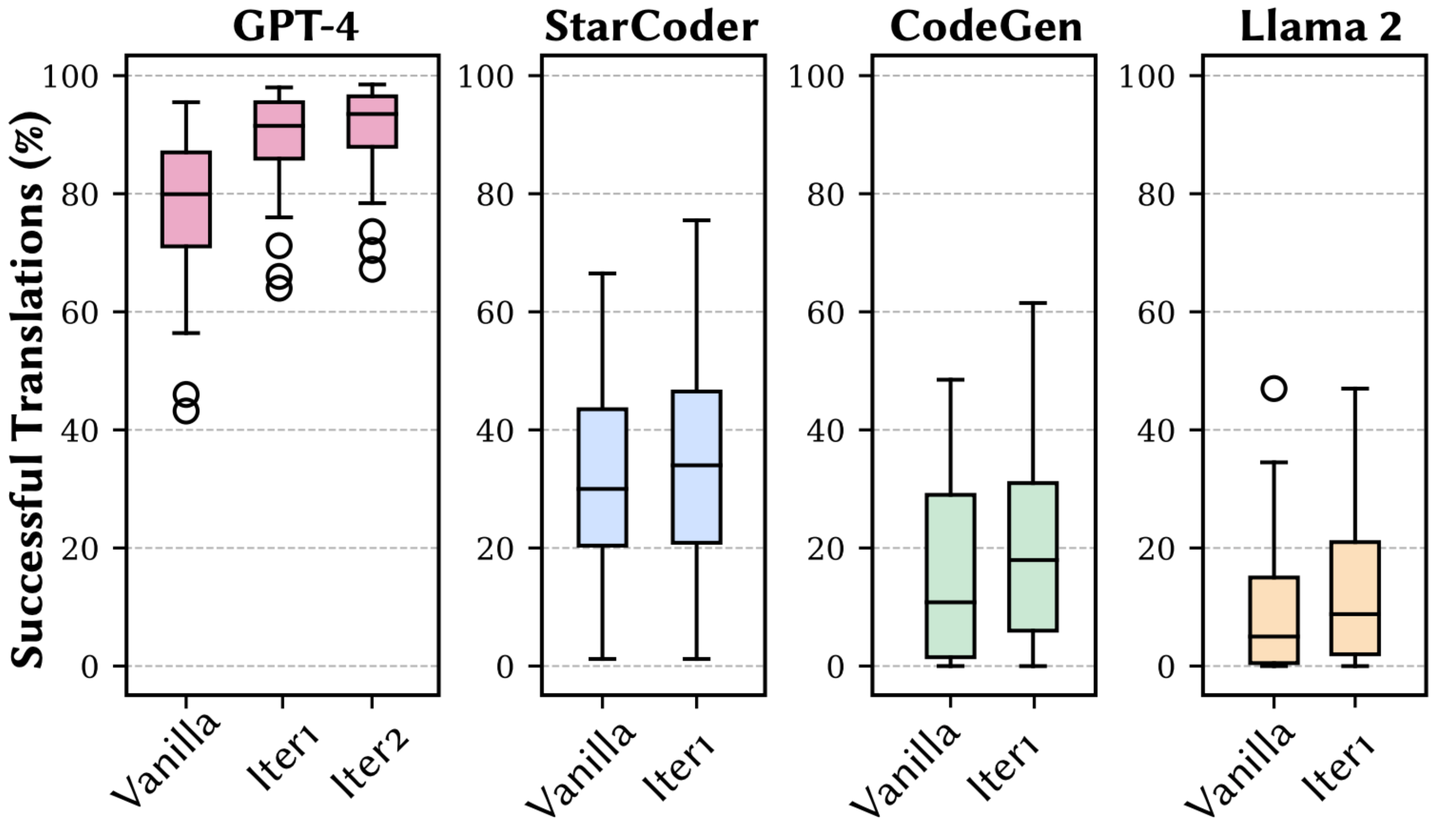
Iterative Translation Bug Mitigation

- We iteratively prompt LLMs with additional feedback to repair unsuccessful cases
  1. Craft prompts based on each effect (runtime error, compile error, test failure)
  2. Stop iteration if overall improvement **<5%**

Iterative Translation Bug Mitigation

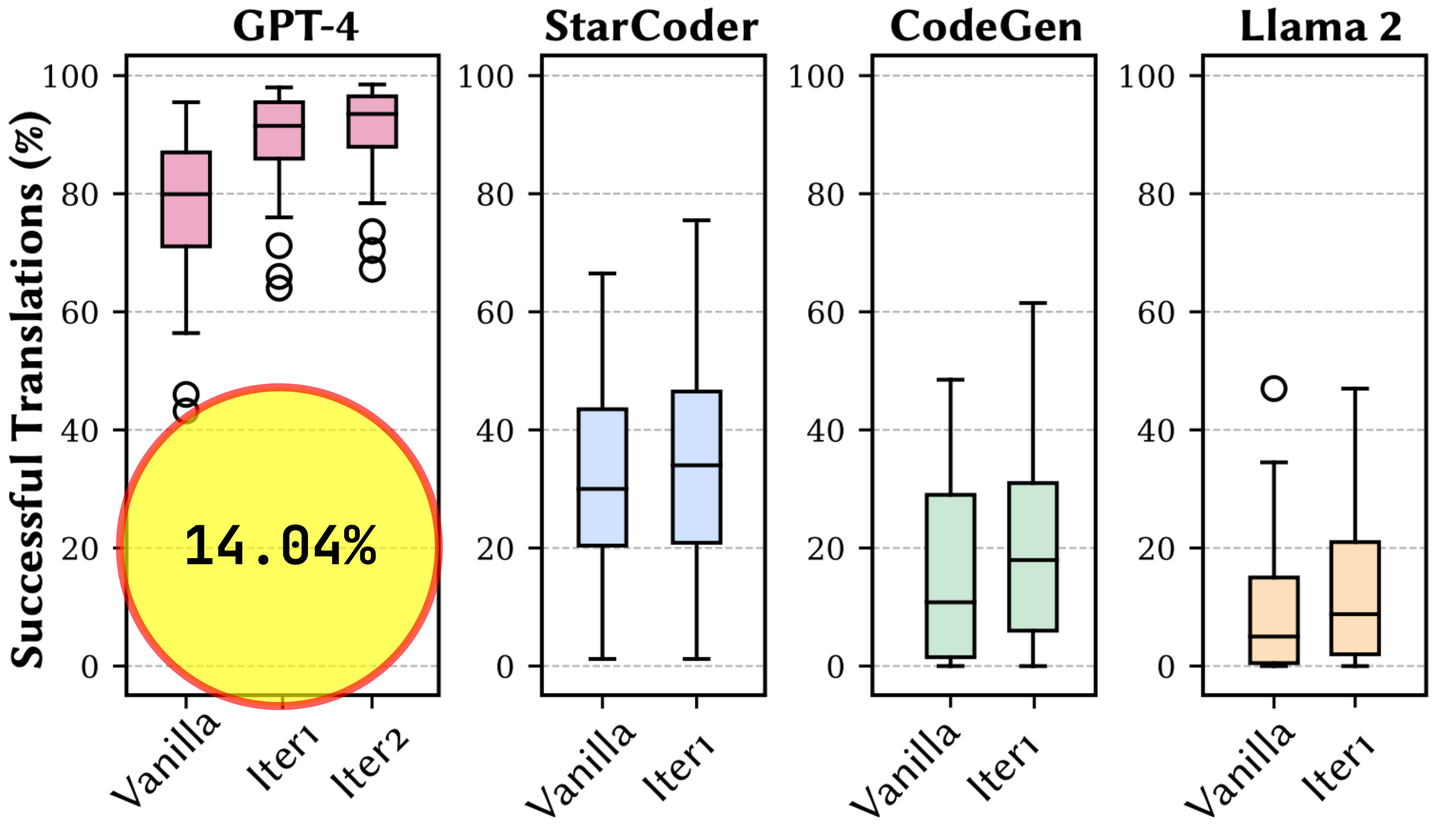
❖ We iteratively prompt LLMs with additional feedback to repair unsuccessful cases

1. Craft prompts based on each effect (runtime error, compile error, test failure)
2. Stop iteration if overall improvement **<5%**



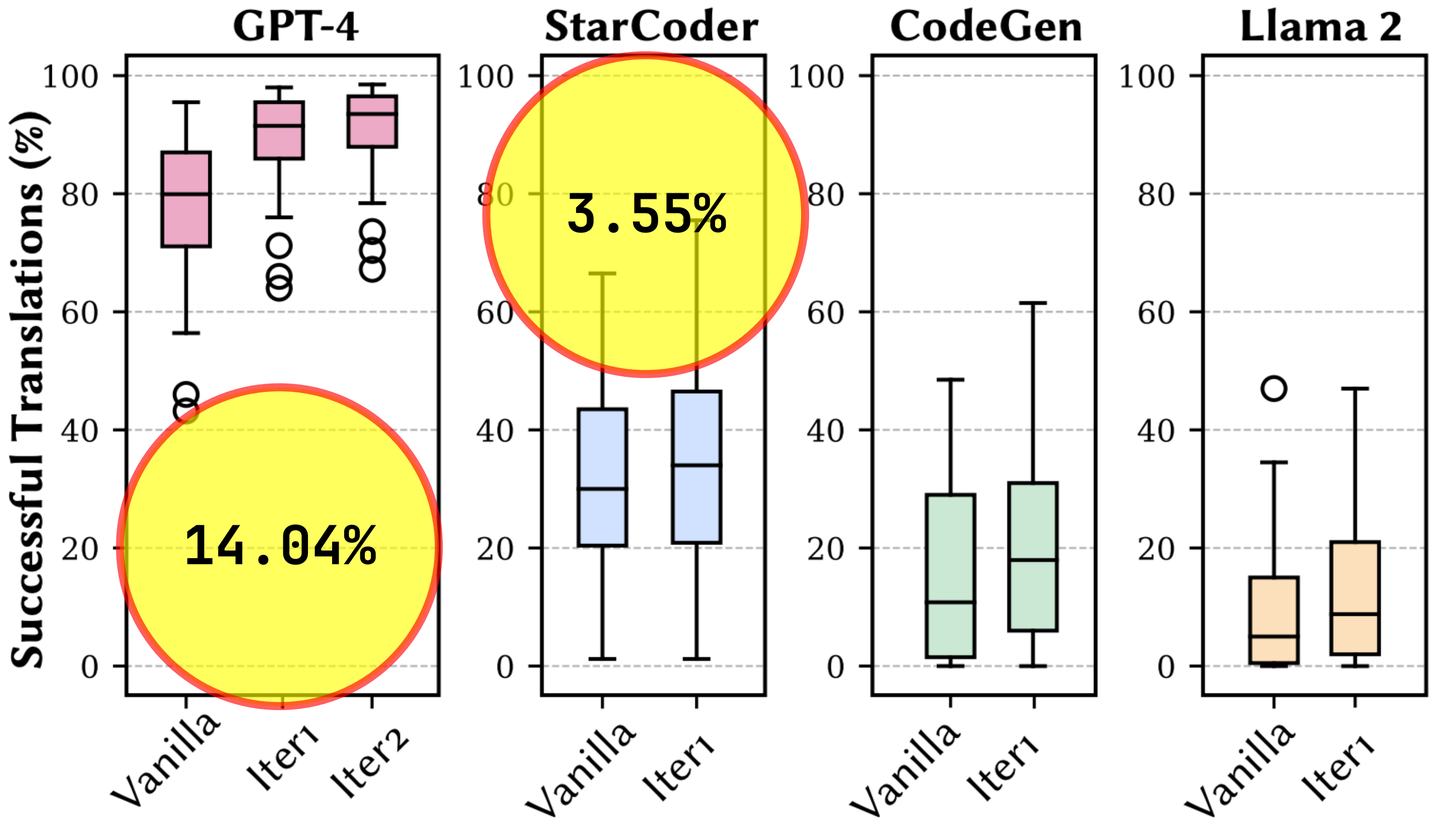
Iterative Translation Bug Mitigation

- We iteratively prompt LLMs with additional feedback to repair unsuccessful cases
  1. Craft prompts based on each effect (runtime error, compile error, test failure)
  2. Stop iteration if overall improvement **<5%**



Iterative Translation Bug Mitigation

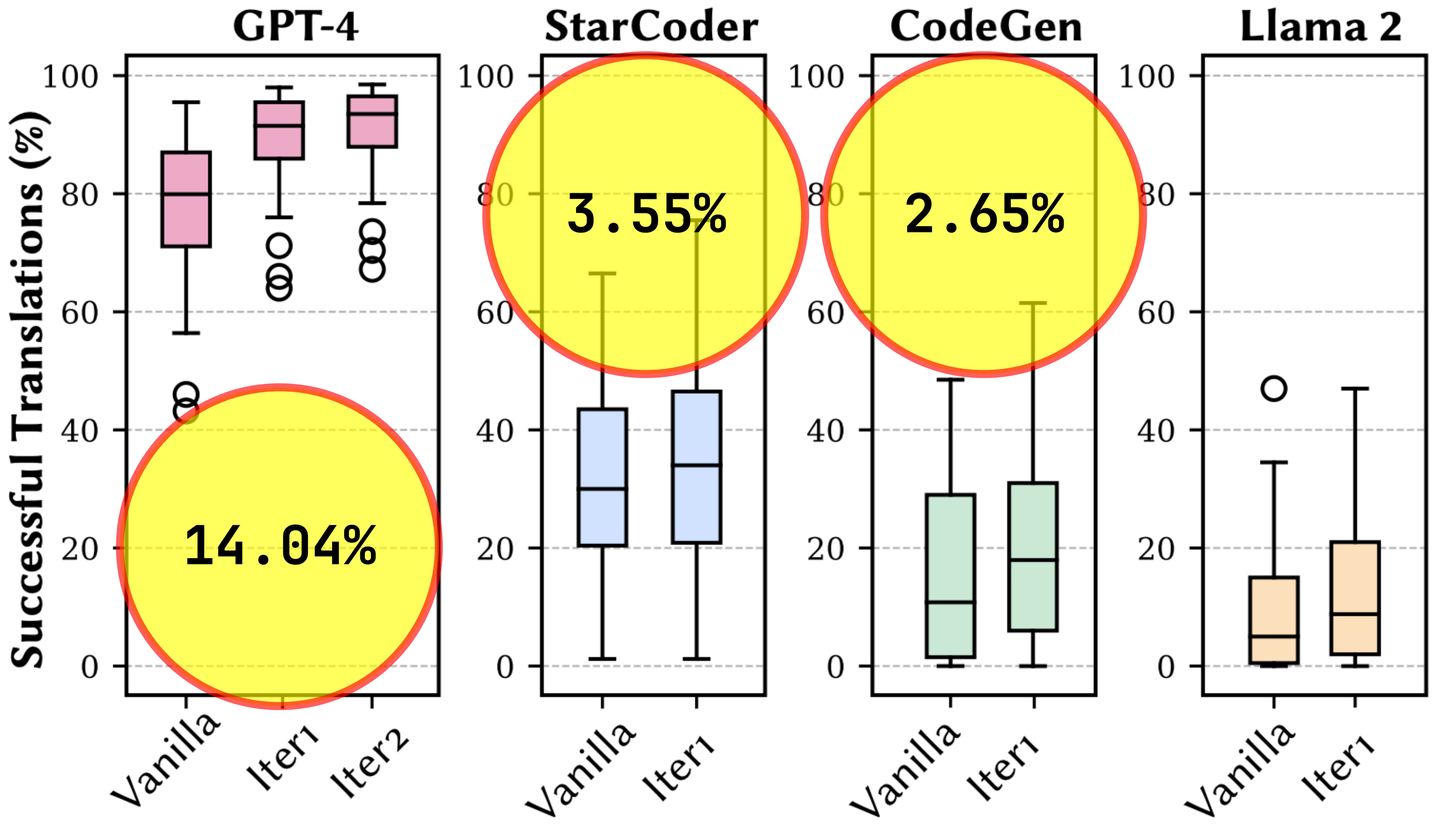
- We iteratively prompt LLMs with additional feedback to repair unsuccessful cases
  1. Craft prompts based on each effect (runtime error, compile error, test failure)
  2. Stop iteration if overall improvement **<5%**



Iterative Translation Bug Mitigation

❖ We iteratively prompt LLMs with additional feedback to repair unsuccessful cases

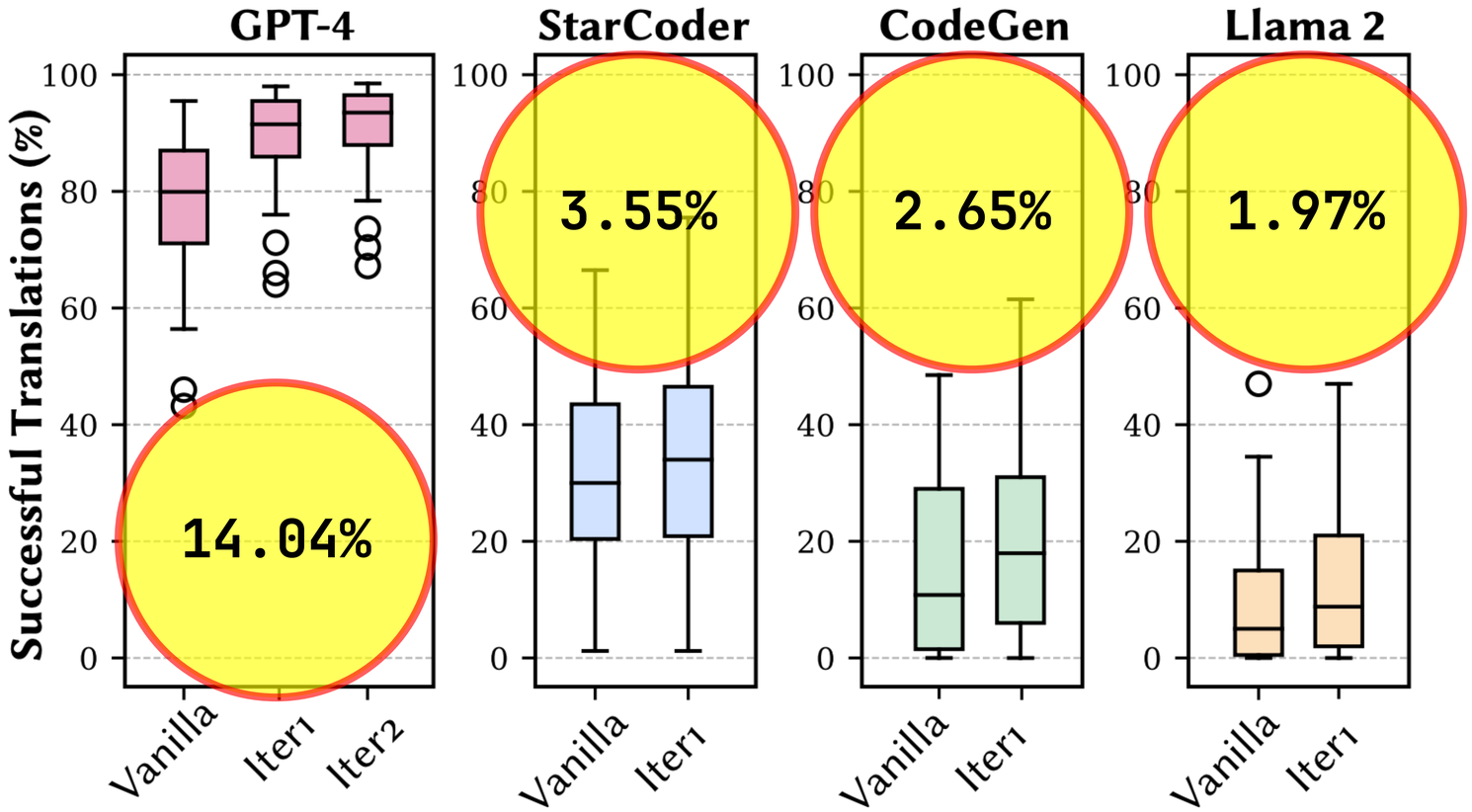
1. Craft prompts based on each effect (runtime error, compile error, test failure)
2. Stop iteration if overall improvement **<5%**



Iterative Translation Bug Mitigation

❖ We iteratively prompt LLMs with additional feedback to repair unsuccessful cases

1. Craft prompts based on each effect (runtime error, compile error, test failure)
2. Stop iteration if overall improvement **<5%**



Comparison with non-LLM-based Techniques

Both approaches provide unique advantages, suggesting an ultimate solution can leverage both LLM and non-LLM (transpiler) techniques

Dataset	SL	TL	CxGO	C2Rust	Java2C#	CodeGen	CodeGeeX	StarCoder	GPT-4	Llama 2	TB-A	TB-V
CodeNet	C	Rust	-	**95.0%**	-	0.0%	0.0%	10.5%	61.0%	0.5%	0.5%	0.0%
	C	Go	62.3%	-	-	33.0%	0.0%	12.5%	**72.5%**	5.0%	5.5%	11.0%
	Java	C#	-	-	0.0%	0.5%	1.0%	26.5%	**49.0%**	1.5%	1.0%	4.0%
AVATAR	Java	C#	-	-	0.0%	0.0%	0.0%	10.4%	**59.2%**	2.0%	0.8%	0.4%

*{S, T} L: {Source, Target} Language, TB-{A, V}: TheBloke-{Airoboros, Vicuna}

Transpiler | **LLM**

Comparison with non-LLM-based Techniques

Both approaches provide unique advantages, suggesting an ultimate solution can leverage both LLM and non-LLM (transpiler) techniques

Dataset	SL	TL	CxGO	C2Rust	Java2C#	CodeGen	CodeGeeX	StarCoder	GPT-4	Llama 2	TB-A	TB-V
CodeNet	C	Rust	-	**95.0%**	-	0.0%	0.0%	10.5%	61.0%	0.5%	0.5%	0.0%
	C	Go	62.3%	-	-	33.0%	0.0%	12.5%	**72.5%**	5.0%	5.5%	11.0%
	Java	C#	-	-	0.0%	0.5%	1.0%	26.5%	**49.0%**	1.5%	1.0%	4.0%
AVATAR	Java	C#	-	-	0.0%	0.0%	0.0%	10.4%	**59.2%**	2.0%	0.8%	0.4%

*{S, T} L: {Source, Target} Language, TB-{A, V}: TheBloke-{Airoboros, Vicuna}

Transpiler | **LLM**

Comparison with non-LLM-based Techniques

Both approaches provide unique advantages, suggesting an ultimate solution can leverage both LLM and non-LLM (transpiler) techniques

Transpilers 🤝 Language Models

Dataset	SL	TL	CxGO	C2Rust	Java2C#	CodeGen	CodeGeeX	StarCoder	GPT-4	Llama 2	TB-A	TB-V
CodeNet	C	Rust	-	**95.0%**	-	0.0%	0.0%	10.5%	61.0%	0.5%	0.5%	0.0%
	C	Go	62.3%	-	-	33.0%	0.0%	12.5%	**72.5%**	5.0%	5.5%	11.0%
	Java	C#	-	-	0.0%	0.5%	1.0%	26.5%	**49.0%**	1.5%	1.0%	4.0%
AVATAR	Java	C#	-	-	0.0%	0.0%	0.0%	10.4%	**59.2%**	2.0%	0.8%	0.4%

*{S, T} L: {Source, Target} Language, TB-{A, V}: TheBloke-{Airoboros, Vicuna}

Transpiler | **LLM**

Unsafety of Transpilers

C2Rust translation is unsafe and mostly depends on external libraries

Unsafety of Transpilers

C2Rust translation is unsafe and mostly depends on external libraries

```rust
fn main() {
    let mut x = String::new();
    let mut y = String::new();
    io::stdin().read_line(&mut x).expect("Failed to read line");
    io::stdin().read_line(&mut y).expect("Failed to read line");
    let x: i32 = x.trim().parse().expect("Please type a number!");
    let y: i32 = y.trim().parse().expect("Please type a number!");
    println!("{} {}", x*y, 2*x + 2*y); }
```

(a) C Code Translated to Rust by GPT-4

```rust
#![allow(dead_code, mutable_transmutes, unused_assignments, unused_mut)]
extern "C" {                                                              1
    fn printf(_: *const libc::c_char, _: ...) -> libc::c_int;             2
    fn scanf(_: *const libc::c_char, _: ...) -> libc::c_int; }            2
unsafe fn main_0() -> libc::c_int {                                       3
    let mut x: libc::c_int = 0;                                           4
    let mut y: libc::c_int = 0;                                           4
    x = 0 as libc::c_int;
    y = 0 as libc::c_int;
    scanf(b"%d\0" as *const u8 as *const libc::c_char,                    5
          &mut x as *mut libc::c_int);                                    5
    scanf(b"%d\0" as *const u8 as *const libc::c_char,
          &mut y as *mut libc::c_int);
    printf(b"%d %d\n\0" as *const u8 as *const libc::c_char,
           x * y, 2 as libc::c_int * x + 2 as libc::c_int * y,);
    return 0 as libc::c_int; }
```

(b) C Code Transpiled with C2Rust with annotations on safety issues.

Unsafety of Transpilers

C2Rust translation is unsafe and mostly depends on external libraries

```rust
fn main() {
    let mut x = String::new();
    let mut y = String::new();
    io::stdin().read_line(&mut x).expect("Failed to read line");
    io::stdin().read_line(&mut y).expect("Failed to read line");
    let x: i32 = x.trim().parse().expect("Please type a number!");
    let y: i32 = y.trim().parse().expect("Please type a number!");
    println!("{} {}", x*y, 2*x + 2*y); }
```

(a) C Code Translated to Rust by GPT-4

```rust
#![allow(dead_code, mutable_transmutes, unused_assignments, unused_mut)]
extern "C" {
        fn printf(_: *const libc::c_char, _: ...) -> libc::c_int;
        fn scanf(_: *const libc::c_char, _: ...) -> libc::c_int; }
unsafe fn main_0() -> libc::c_int {
        let mut x: libc::c_int = 0;
        let mut y: libc::c_int = 0;
        x = 0 as libc::c_int;
        y = 0 as libc::c_int;
        scanf(b"%d\0" as *const u8 as *const libc::c_char,
              &mut x as *mut libc::c_int);
        scanf(b"%d\0" as *const u8 as *const libc::c_char,
              &mut y as *mut libc::c_int);
        printf(b"%d %d\n\0" as *const u8 as *const libc::c_char,
               x * y, 2 as libc::c_int * x + 2 as libc::c_int * y,);
        return 0 as libc::c_int; }
```

1 2 3 4 5

(b) C Code Transpiled with C2Rust with annotations on safety issues.

Compiler directives introduce dead code and cause race condition

Unsafety of Transpilers

C2Rust translation is unsafe and mostly depends on external libraries

```rust
fn main() {
    let mut x = String::new();
    let mut y = String::new();
    io::stdin().read_line(&mut x).expect("Failed to read line");
    io::stdin().read_line(&mut y).expect("Failed to read line");
    let x: i32 = x.trim().parse().expect("Please type a number!");
    let y: i32 = y.trim().parse().expect("Please type a number!");
    println!("{} {}", x*y, 2*x + 2*y); }
```

(a) C Code Translated to Rust by GPT-4

```rust
#![allow(dead_code, mutable_transmutes, unused_assignments, unused_mut)]
extern "C" {                                                              1
    fn printf(_: *const libc::c_char, _: ...) -> libc::c_int;             2
    fn scanf(_: *const libc::c_char, _: ...) -> libc::c_int; }
unsafe fn main_0() -> libc::c_int {                                       3
    let mut x: libc::c_int = 0;                                           4
    let mut y: libc::c_int = 0;
    x = 0 as libc::c_int;
    y = 0 as libc::c_int;
    scanf(b"%d\0" as *const u8 as *const libc::c_char,                    5
          &mut x as *mut libc::c_int);
    scanf(b"%d\0" as *const u8 as *const libc::c_char,
          &mut y as *mut libc::c_int);
    printf(b"%d %d\n\0" as *const u8 as *const libc::c_char,
           x * y, 2 as libc::c_int * x + 2 as libc::c_int * y,);
    return 0 as libc::c_int; }
```

(b) C Code Transpiled with C2Rust with annotations on safety issues.

Foreign functions may not conform to Rust's safety guarantees, e.g. using "scanf" without buffer size

Compiler directives introduce dead code and cause race condition

Unsafety of Transpilers

C2Rust translation is unsafe and mostly depends on external libraries

```rust
fn main() {
    let mut x = String::new();
    let mut y = String::new();
    io::stdin().read_line(&mut x).expect("Failed to read line");
    io::stdin().read_line(&mut y).expect("Failed to read line");
    let x: i32 = x.trim().parse().expect("Please type a number!");
    let y: i32 = y.trim().parse().expect("Please type a number!");
    println!("{} {}", x*y, 2*x + 2*y); }
```

(a) C Code Translated to Rust by GPT-4

```rust
#![allow(dead_code, mutable_transmutes, unused_assignments, unused_mut)]
extern "C" {                                                              1
    fn printf(_: *const libc::c_char, _: ...) -> libc::c_int;             2
    fn scanf(_: *const libc::c_char, _: ...) -> libc::c_int; }
unsafe fn main_0() -> libc::c_int {                                       3
    let mut x: libc::c_int = 0;                                           4
    let mut y: libc::c_int = 0;
    x = 0 as libc::c_int;
    y = 0 as libc::c_int;
    scanf(b"%d\0" as *const u8 as *const libc::c_char,                    5
          &mut x as *mut libc::c_int);
    scanf(b"%d\0" as *const u8 as *const libc::c_char,
          &mut y as *mut libc::c_int);
    printf(b"%d %d\n\0" as *const u8 as *const libc::c_char,
           x * y, 2 as libc::c_int * x + 2 as libc::c_int * y,);
    return 0 as libc::c_int; }
```

(b) C Code Transpiled with C2Rust with annotations on safety issues.

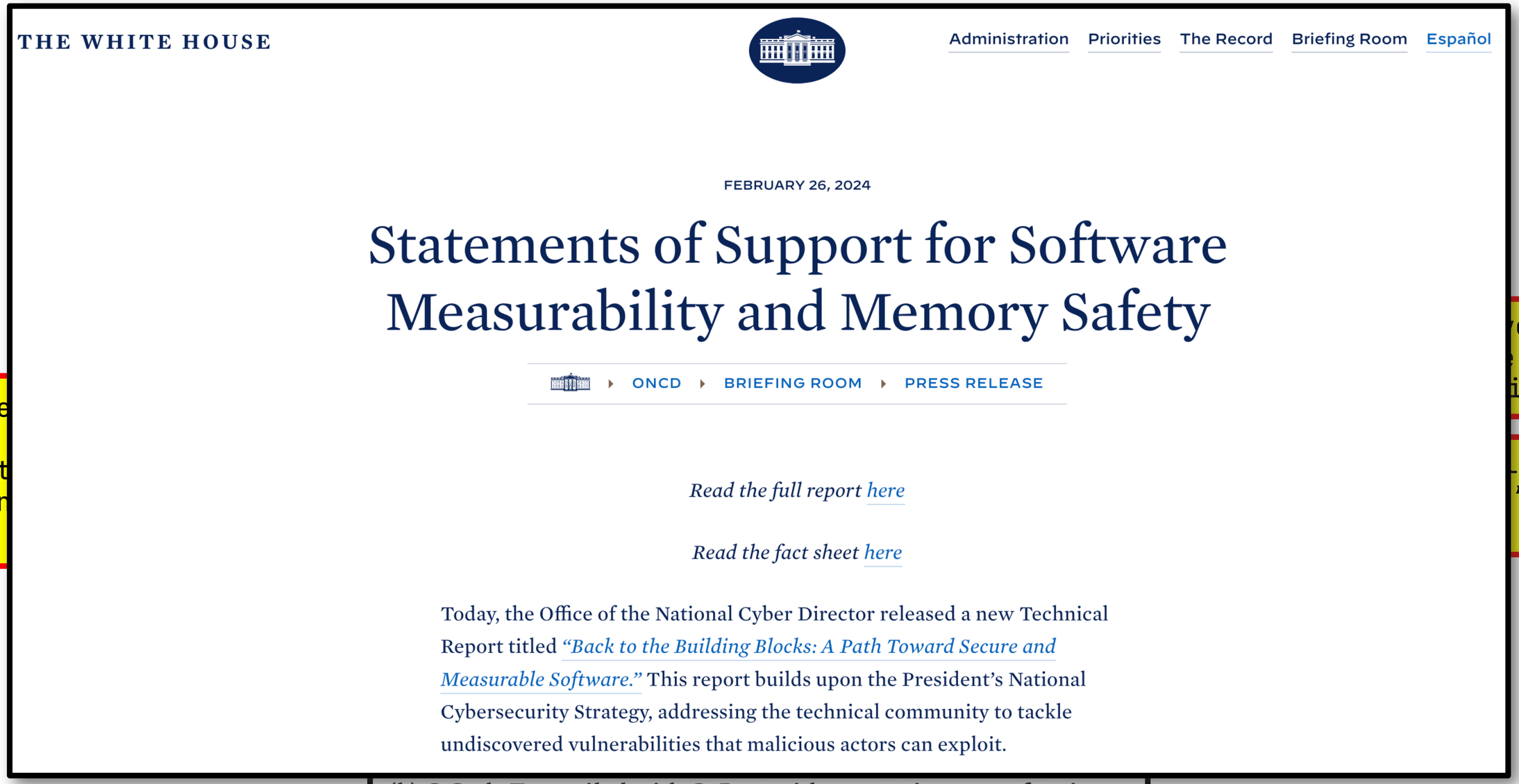
Foreign functions may not conform to Rust's safety guarantees, e.g. using "scanf" without buffer size

Compiler directives introduce dead code and cause race condition

Usage of "unsafe block" circumvents Rust's safety checks

Unsafety of Transpilers

https://www.whitehouse.gov/oncd/briefing-room/2024/02/26/memory-safety-statements-of-support/



THE WHITE HOUSE

Administration Priorities The Record Briefing Room Español

FEBRUARY 26, 2024

Statements of Support for Software Measurability and Memory Safety

ONCD ▸ BRIEFING ROOM ▸ PRESS RELEASE

Read the full report here

Read the fact sheet here

Today, the Office of the National Cyber Director released a new Technical Report titled *"Back to the Building Blocks: A Path Toward Secure and Measurable Software."* This report builds upon the President's National Cybersecurity Strategy, addressing the technical community to tackle undiscovered vulnerabilities that malicious actors can exploit.

(b) C Code Transpiled with C2Rust with annotations on safety issues.

Concluding Remarks

❖ We investigate the effectiveness of **7** LLMs across **5** datasets in code translation

Concluding Remarks

- We investigate the effectiveness of **7** LLMs across **5** datasets in code translation
- We identify and open-source **15** root causes of translation bugs

Concluding Remarks

- We investigate the effectiveness of **7** LLMs across **5** datasets in code translation
- We identify and open-source **15** root causes of translation bugs
- We evaluate **3** non-LLM-based techniques on our benchmarks

Concluding Remarks

- We investigate the effectiveness of **7** LLMs across **5** datasets in code translation
- We identify and open-source **15** root causes of translation bugs
- We evaluate **3** non-LLM-based techniques on our benchmarks
- To improve LLMs, we propose an iterative prompt crafting technique

Concluding Remarks

- We investigate the effectiveness of **7** LLMs across **5** datasets in code translation
- We identify and open-source **15** root causes of translation bugs
- We evaluate **3** non-LLM-based techniques on our benchmarks
- To improve LLMs, we propose an iterative prompt crafting technique
- Future Work:
  - **`Neuro-symbolic`** approach to enable **`repository-level`** code translation

Concluding Remarks

- We investigate the effectiveness of **7** LLMs across **5** datasets in code translation
- We identify and open-source **15** root causes of translation bugs
- We evaluate **3** non-LLM-based techniques on our benchmarks
- To improve LLMs, we propose an iterative prompt crafting technique
- Future Work:
  - **`Neuro-symbolic`** approach to enable **`repository-level`** code translation

https://github.com/Intelligent-CAT-Lab/PLTranslationEmpirical

Please Check and Contribute to our Leaderboard

Leaderboard: codetlingua.github.io